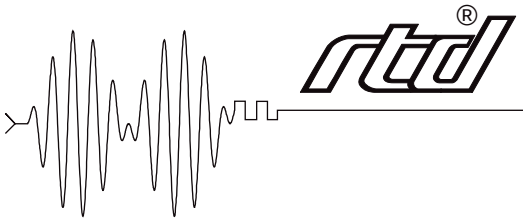
# DIO4806
# User's Manual



**Real Time Devices, Inc.**

*"Accessing the Analog World"®*

Publication No. 4806-9552

# DIO4806

# User's Manual

**REAL TIME DEVICES, INC.**
Post Office Box 906
State College, Pennsylvania 16804 USA
Phone: (814) 234-8087
FAX: (814) 234-5218

Published by
Real Time Devices, Inc.
P.O. Box 906
State College, PA  16804 USA

Copyright © 1995 by Real Time Devices, Inc.
All rights reserved

Printed in U.S.A.

# Table of Contents

**INTRODUCTION** ..... ***i*-1**

Digital I/O ..... *i*-3
8254 Timer/Counters ..... *i*-3
What Comes With Your Board ..... *i*-3
Board Accessories ..... *i*-3
Application Software and Drivers ..... *i*-3
Hardware Accessories ..... *i*-4
Using This Manual ..... *i*-4
When You Need Help ..... *i*-4

**CHAPTER 1 — BOARD SETTINGS** ..... **1-1**

Factory-Configured Switch and Jumper Settings ..... 1-3
P19 — 8254 Clock and Gate Source Select (Factory Settings: See Figure 1-2) ..... 1-5
P20 through P22 — Strobe Input Enable (Factory Setting: Disabled) ..... 1-6
S1 — Base Address (Factory Setting: 300 hex (768 decimal)) ..... 1-7
P7 through P12, Pull-up/Pull-down Resistors on Digital I/O Lines ..... 1-8

**CHAPTER 2 — BOARD INSTALLATION** ..... **2-1**

Board Installation ..... 2-3
External I/O Connections ..... 2-3
Connecting the Digital I/O ..... 2-4
Connecting the Timer/Counter I/O ..... 2-4
Connecting the External Interrupt ..... 2-4
Running the 4806DIAG Diagnostics Program ..... 2-4

**CHAPTER 3 — HARDWARE DESCRIPTION** ..... **3-1**

Digital I/O ..... 3-3
Timer/Counters ..... 3-3

**CHAPTER 4 — I/O MAPPING** ..... **4-1**

Defining the I/O Map ..... 4-3
BA + 0: Digital I/O Port 0, Bit Programmable Port (Read/Write) ..... 4-4
BA + 1: Digital I/O Port 1, Byte Programmable Port (Read/Write) ..... 4-4
BA + 2: Read/Program Port 0 Direction/Mask/Compare Registers (Read/Write) ..... 4-4
BA + 3: Read Digital I/O Status/Program Digital Mode (Read/Write) ..... 4-5
BA + 4: Digital I/O Port 2, Bit Programmable Port (Read/Write) ..... 4-6
BA + 5: Digital I/O Port 3, Byte Programmable Port (Read/Write) ..... 4-6
BA + 6: Read/Program Port 2 Direction/Mask/Compare Registers (Read/Write) ..... 4-6
BA + 7: Read Digital I/O Status/Program Digital Mode (Read/Write) ..... 4-7
BA + 8: Digital I/O Port 4, Bit Programmable Port (Read/Write) ..... 4-8
BA + 9: Digital I/O Port 5, Byte Programmable Port (Read/Write) ..... 4-8
BA + 10: Read/Program Port 4 Direction/Mask/Compare Registers (Read/Write) ..... 4-8
BA + 11: Read Digital I/O Status/Program Digital Mode (Read/Write) ..... 4-9
BA + 12: 8254 Timer/Counter 0 (Read/Write) ..... 4-10
BA + 13: 8254 Timer/Counter 1 (Read/Write) ..... 4-10
BA + 14: 8254 Timer/Counter 2 (Read/Write) ..... 4-10
BA + 15: 8254 Timer/Counter Control Word (Write Only) ..... 4-10
BA + 16: Clear IRQ/IRQ Enable (Read/Write) ..... 4-10

BA + 17: IRQ Status (Read Only) ..... 4-11
BA + 18: Reserved ..... 4-11
BA + 19: IRQ Channel/Source Select (Read/Write) ..... 4-11
Programming the DIO4806 ..... 4-12
Clearing and Setting Bits in a Port ..... 4-12

**CHAPTER 5 — DIGITAL I/O ..... 5-1**

Ports 0, 2, and 4: Bit Programmable Digital I/O ..... 5-3
Advanced Digital Interrupts: Mask and Compare Registers ..... 5-3
Ports 1, 3, and 5: Port Programmable Digital I/O ..... 5-3
Resetting the Digital Circuitry ..... 5-3
Strobing Data into Ports 0, 2, and 4 ..... 5-3

**CHAPTER 6 — TIMER/COUNTERS ..... 6-1**

**CHAPTER 7 — INTERRUPTS ..... 7-1**

Software Programmable Interrupts ..... 7-3
Advanced Digital Interrupts ..... 7-3
Event Mode ..... 7-3
Match Mode ..... 7-3
Sampling Digital Lines for Change of State ..... 7-3
Selecting the Interrupt Channel ..... 7-4
Basic Programming For Interrupt Handling ..... 7-5
What Is an Interrupt? ..... 7-5
Interrupt Request Lines ..... 7-5
8259 Programmable Interrupt Controller ..... 7-5
Interrupt Mask Register (IMR) ..... 7-5
End-of-Interrupt (EOI) Command ..... 7-5
What Exactly Happens When an Interrupt Occurs? ..... 7-6
Using Interrupts in Your Programs ..... 7-6
Writing an Interrupt Service Routine (ISR) ..... 7-6
Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector ..... 7-8
Restoring the Startup IMR and Interrupt Vector ..... 7-8
Common Interrupt Mistakes ..... 7-8

**APPENDIX A — DIO4806 SPECIFICATIONS ..... A-1**

**APPENDIX B — CONNECTOR PIN ASSIGNMENTS ..... B-1**

**APPENDIX C — COMPONENT DATA SHEETS ..... C-1**

**APPENDIX D — WARRANTY ..... D-1**

# List of Illustrations


1-1 Board Layout Showing Factory-Configured Settings ..... 1-4
1-2 8254 Clock and Gate Sources Jumpers, P19 ..... 1-5
1-3 8254 Circuit Diagram ..... 1-6
1-4 Port 0 Strobe Input Enable Jumper, P20 ..... 1-6
1-5 Base Address Switch, S1 ..... 1-7
1-6 Port 0 Pull-up/Pull-down Resistor Connections, P7 ..... 1-8
2-1 P2 and P3 50-pin I/O Connector Pin Assignments ..... 2-3
2-1 P6 20-pin I/O Connector Pin Assignments ..... 2-4
3-1 DIO4806 Block Diagram ..... 3-3
3-2 Timer/Counter Circuit Block Diagram ..... 3-4
6-1 8254 Timer/Counter Circuit Block Diagram ..... 6-3
7-1 Digital Interrupt Timing Diagram ..... 7-3

# INTRODUCTION

The DIO4806 opto-22 compatible digital I/O Advanced Industrial Control board is for use in an IBM PC-compatible computer. Installed in a single full-size slot, the DIO4806 features:

- 24 bit programmable digital I/O lines with Advanced Digital Interrupt modes plus 24 port programmable digital I/O lines,
- Pull-up/pull-down resistors on each bit,
- Three 16-bit timer/counters and on-board 8 MHz clock,
- Direct connection to opto-22 I/O system modules,
- Operation from single +5V supply,
- DOS example programs with source code in QuickBASIC and C,
- Diagnostics software.

The following paragraphs briefly describe the major functions of the board. A detailed discussion of board functions is included in subsequent chapters.

## Digital I/O

The DIO4806 has 48 buffered TTL/CMOS digital I/O lines which are grouped into six 8-bit ports, Port 0 through Port 5. Ports 0, 2, and 4 are bit programmable lines and Ports 1, 3, and 5 are port programmable lines. The bit programmable lines support RTD's two Advanced Digital Interrupt modes. An interrupt can be generated when any bit changes value (event interrupt), or when the lines match a programmed value (match interrupt). For either mode, masking can be used to monitor selected lines.

Bit configurable pull-up or pull-down resistors are provided for all 48 lines. Instructions for activating these pull-up/pull-down resistors are given at the end of Chapter 1, *Board Settings*.

## 8254 Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8 MHz timer/counters to support a wide range of user timing and counting functions.

## What Comes With Your Board

You receive the following items in your board package:

- DIO4806 opto-22 compatible digital control board
- DOS example programs in QuickBASIC and C with source code & diagnostics software
- User's manual

If any item is missing or damaged, please call Real Time Devices' Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

## Board Accessories

In addition to the items included in your board package, Real Time Devices offers a full line of software and hardware accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your board's application.

### Hardware Accessories

Hardware accessories for the DIO4806 include the DOP series optoisolated digital input front-end boards, the DMR series mechanical relay output front-end boards, the TB50 terminal board and XB50 prototype/terminal board for easy signal access and prototype development, and XO50 ribbon cable assembly for interconnection to an opto-22 rack.

## Using This Manual

This manual is intended to help you install your new board and get it running quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in the most complex applications. We assume that you already have an understanding of data acquisition principles and that you can customize the example software or write your own application programs.

## When You Need Help

This manual and the example programs in the software package included with your board provide enough information to properly use all of the board's features. If you have any problems installing or using this Advanced Industrial Control board, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

# CHAPTER 1

## BOARD SETTINGS

The DIO4806 has jumper and switch settings you can change if necessary for your application. The board is factory-configured as listed in the table and shown on the layout diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the board in your computer.

Also note that by setting the jumpers as desired on header connectors P7 through P12, you can configure each digital I/O line to be pulled up or pulled down. This procedure is explained at the end of this chapter.

## Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switch on the DIO4806. Figure 1-1 shows the board layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your module in your system.

**Table 1-1: Factory Settings**

| Switch/ Jumper | Function Controlled | Factory Settings (Jumpers Installed) |
|---|---|---|
| P7 | Activates pull-up/ pull-down resistors on Port 0 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P8 | Activates pull-up/ pull-down resistors on Port 1 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P9 | Activates pull-up/ pull-down resistors on Port 2 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P10 | Activates pull-up/ pull-down resistors on Port 3 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P11 | Activates pull-up/ pull-down resistors on Port 4 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P12 | Activates pull-up/ pull-down resistors on Port 5 digital I/O lines | All bits pulled up (jumpers installed between COM & V) |
| P13-P18 | Not used | No jumpers installed |
| P19 | Sets the clock and gate sources for the 8254 timer/counter | CLK0: OSC; CLK1: OUTT0 CLK2:OOUT1; GATE2:EXTGATE2 (timer/counters cascaded) |
| P20 | Enables and connects a strobe input to Port 0 through the selected EXT INT (external interrupt) line | DISABLE (disabled) |
| P21 | Enables and connects a strobe input to Port 2 through the selected EXT INT (external interrupt) line | DISABLE (disabled) |
| P22 | Enables and connects a strobe input to Port 4 through the selected EXT INT (external interrupt) line | DISABLE (disabled) |
| P23-P25 | Not used | No jumpers installed |
| S1 | Sets the base address | 300 hex (768 decimal) |

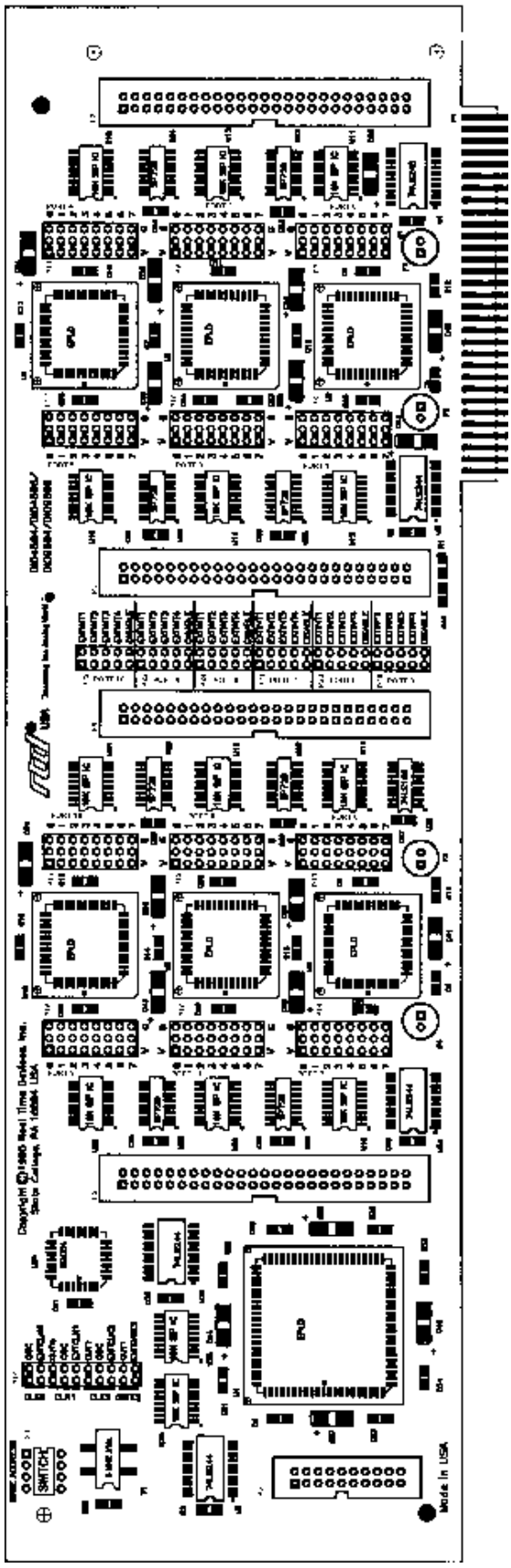


Fig. 1-1 — Board Layout Showing Factory Configured Settings

**P19 — 8254 Clock and Gate Source Select (Factory Settings: See Figure 1-2)**

This header connector, shown in Figure 1-2, lets you select the clock sources for the three 8254 16-bit timer/counters. Figure 1-3 shows a block diagram of the timer/counter circuitry to help you in making these connections.

The clock source for Counter 0 is selected by placing a jumper on one of the two top pairs of pins on the header, OSC or EXTCLK0. OSC is the on-board 8 MHz clock, and EXTCLK0 is an external clock source which can be connected through I/O connector P6, pin 1. Counter 1 has three clock sources: OUT0, which cascades it to Counter 0; OSC, which is the on-board 8 MHz clock; and EXTCLK1, which is an external clock source connected through I/O connector P6, pin 7. Counter 2 has three clock sources: OUT1, which cascades it to Counter 1; OSC, which is the on-board 8 MHz clock; and EXTCLK2, which is an external clock source connected through I/O connector P6, pin 13.

The gate of Counter 2 can be connected to the output of Counter 1 (OUT1) or to an external gate source (EXTGATE2) connected through I/O connector P6, pin 15. When no external gate source is connected, this line is tied high.

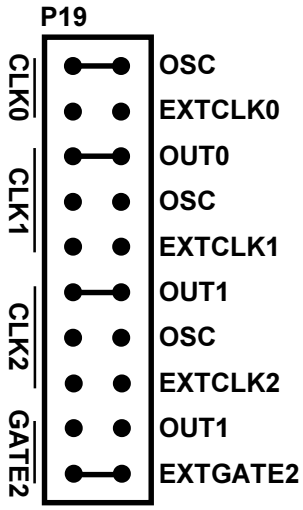



Fig. 1-2 — 8254 Clock and Gate Sources Jumpers, P19

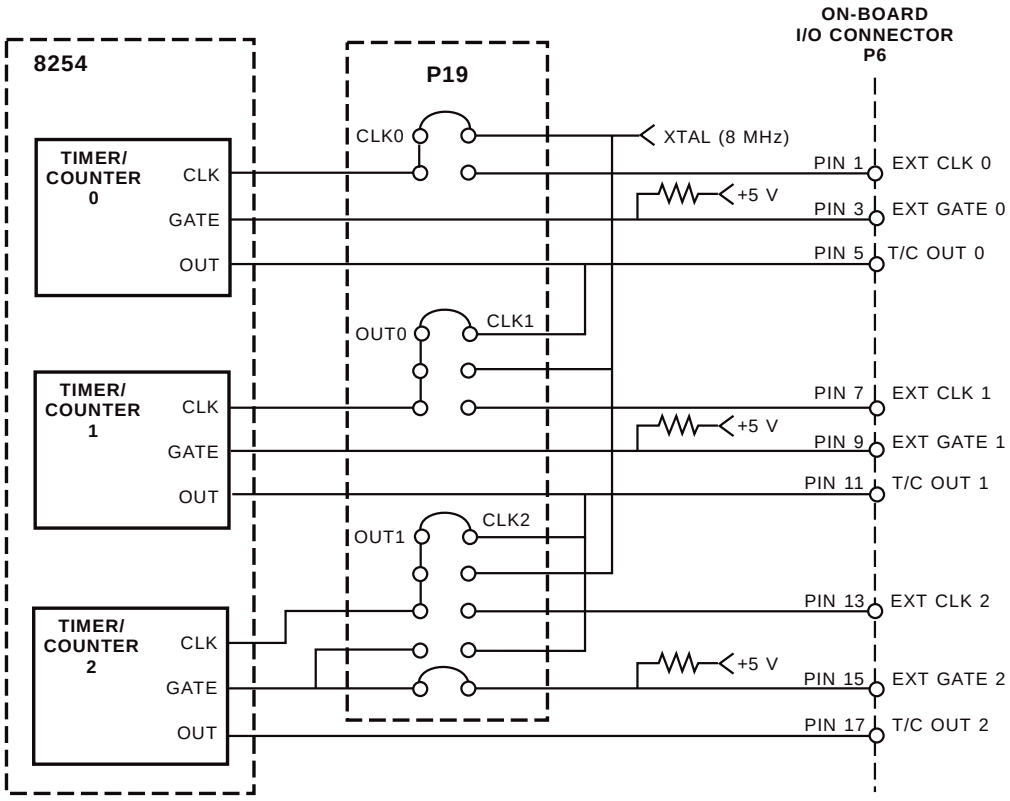



Fig. 1-3 — 8254 Circuit Diagram

**P20 through P22 — Strobe Input Enable (Factory Setting: Disabled)**

These header connectors connect an external signal (through P2-2, EXTINT1; or P3-2, EXTINT2) to the strobe input of Port 0, Port 2, and/or Port 4. To enable the strobe input on the digital I/O chip of a selected port, you must remove the port's jumper from the DISABLE pins and place it across the desired EXTINT pins. EXTINT3 and EXTINT4 are not connected. Note that multiple ports can be strobed from the same signal. Figure 1-4 shows Port 0, P20. Port 2 is P21, and Port 4 is P22.

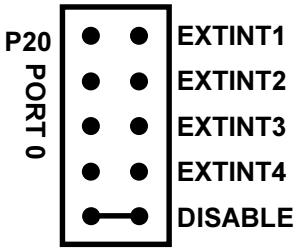



Fig. 1-4a: Strobe Disabled
(Factory Setting)

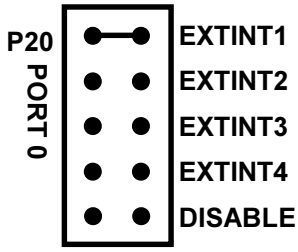



Fig. 1-4b: Strobe Enabled
using EXTINT1

Fig. 1-4 — Port 0 Strobe Input Enable Jumper, P20

**S1 — Base Address (Factory Setting: 300 hex (768 decimal))**

One of the most common causes of failure when you are first trying your board is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the board attempts to use I/O address locations already used by another device, contention results and the board does not work.

To avoid this problem, the DIO4806 has an easily accessible DIP switch, S1, which lets you select any one of 16 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any one of the values listed in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 4) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your board, record the value in the table inside the back cover. Figure 1-5 shows the DIP switch set for a base address of 300 hex (768 decimal).

**Table 1-2 Base Address Switch Settings, S1**

| Base Address Decimal / (Hex) | Switch Setting 4 3 2 1 | Base Address Decimal / (Hex) | Switch Setting 4 3 2 1 |
|---|---|---|---|
| 512 / (200) | 0 0 0 0 | 768 / (300) | 1 0 0 0 |
| 544 / (220) | 0 0 0 1 | 800 / (320) | 1 0 0 1 |
| 576 / (240) | 0 0 1 0 | 832 / (340) | 1 0 1 0 |
| 608 / (260) | 0 0 1 1 | 864 / (360) | 1 0 1 1 |
| 640 / (280) | 0 1 0 0 | 896 / (380) | 1 1 0 0 |
| 672 / (2A0) | 0 1 0 1 | 928 / (3A0) | 1 1 0 1 |
| 704 / (2C0) | 0 1 1 0 | 960 / (3C0) | 1 1 1 0 |
| 736 / (2E0) | 0 1 1 1 | 992 / (3E0) | 1 1 1 1 |
| **0 = closed, 1 = open** | | | |

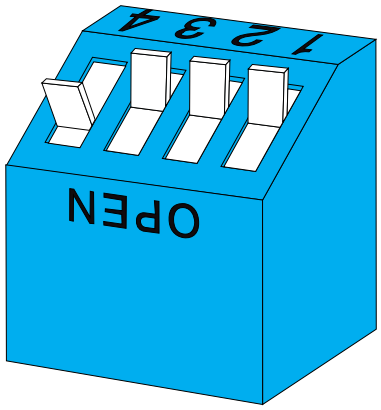



Fig. 1-5 — Base Address Switch, S1

## P7 through P12, Pull-up/Pull-down Resistors on Digital I/O Lines

The DIO4806 has 48 TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. These lines are divided into six 8-bit ports: Ports 0, 2, and 4 with eight individual bit programmable lines each, and Ports 1, 3, and 5 with eight port programmable lines each. You can connect pull-up or pull-down resistors to any or all of these lines on a bit by bit basis. You may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull lines down for connection to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high. By pulling these lines down, you can ensure that when the data acquisition system is first turned on, the motors will not switch on before the port is initialized.

Pull-up/pull-down resistors have been factory installed on the board, and jumpers have been installed in the pull-up position on P7 through P12 for all 48 I/O lines. Each port and bit is labeled on the module. P7 connects to the resistors for Port 0, P8 connects to the resistors for Port 1, and so on. The pins are labeled G (for ground) on one end and V (for +5V) on the other end. The middle pin is common. Figure 1-6 shows P7 with the factory-installed jumpers placed between the common pin (middle pin of the three) and the V pin. For pull-downs, install the jumper across the common pin (middle pin) and G pin. To disable the pull-up/pull-down resistor, remove the jumper.

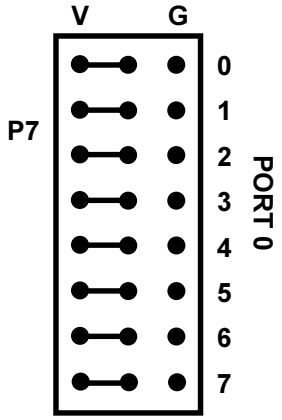



Fig. 1-6 — Port 0 Pull-up/Pull-down Resistor Connections, P7

# CHAPTER 2

## BOARD INSTALLATION

The DIO4806 is easy to install in your PC. This chapter tells you step-by-step how to install and connect the board.

After you have installed the board and made all of your connections, you can turn your system on and run the 4806DIAG board diagnostics program included on your example software disk to verify that your board is working.

## Board Installation

Keep the board in its antistatic bag until you are ready to install it in your computer. When removing it from the bag, hold the board at the edges and do not touch the components or connectors.

Before installing the board in your computer, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

To install the board:

1. Turn OFF the power to your computer.
2. Remove the top cover of the computer housing (refer to your owner's manual if you do not already know how to do this).
3. Select any unused full-size expansion slot and remove the slot bracket.
4. Touch the metal housing of the computer to discharge any static buildup and then remove the board from its antistatic bag.
5. Holding the board by its edges, orient it so that its card edge (bus) connector lines up with the expansion slot connector in the bottom of the selected expansion slot.
6. After carefully positioning the board in the expansion slot so that the card edge connector is resting on the computer's bus connector, gently and evenly press down on the board until it is secured in the slot.

   NOTE: Do not force the board into the slot. If the board does not slide into place, remove it and try again. Wiggling the board or exerting too much pressure can result in damage to the board or to the computer.
7. After the board is installed, secure the slot bracket back into place and put the cover back on your computer. The board is now ready to be connected via the external I/O connector at the rear panel of your computer.

## External I/O Connections

Figure 2-1 shows I/O connector pinouts for the digital I/O connectors, P2 and P3. Figure 2-2 shows the I/O connector pinout for the timer/counter connector, P6. Refer to these diagrams as you make your I/O connections.

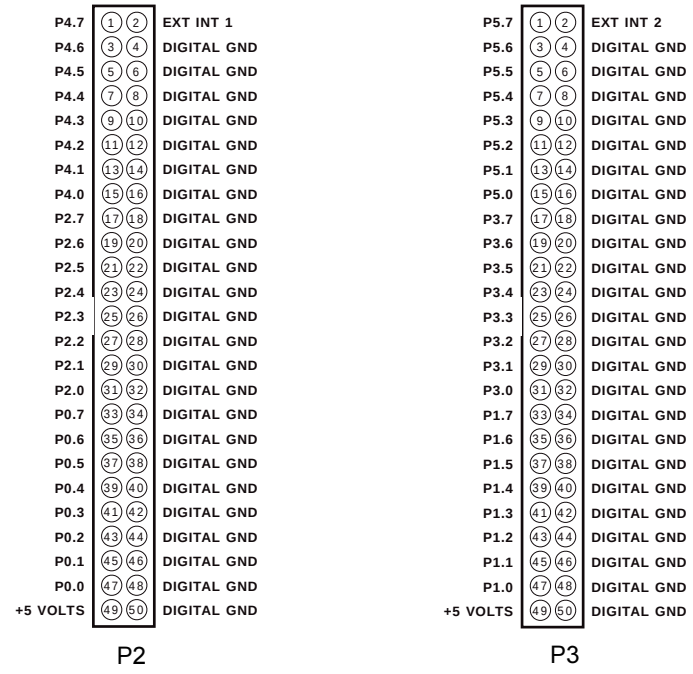

P4.7 1 2 EXT INT 1
P4.6 3 4 DIGITAL GND
P4.5 5 6 DIGITAL GND
P4.4 7 8 DIGITAL GND
P4.3 9 10 DIGITAL GND
P4.2 11 12 DIGITAL GND
P4.1 13 14 DIGITAL GND
P4.0 15 16 DIGITAL GND
P2.7 17 18 DIGITAL GND
P2.6 19 20 DIGITAL GND
P2.5 21 22 DIGITAL GND
P2.4 23 24 DIGITAL GND
P2.3 25 26 DIGITAL GND
P2.2 27 28 DIGITAL GND
P2.1 29 30 DIGITAL GND
P2.0 31 32 DIGITAL GND
P0.7 33 34 DIGITAL GND
P0.6 35 36 DIGITAL GND
P0.5 37 38 DIGITAL GND
P0.4 39 40 DIGITAL GND
P0.3 41 42 DIGITAL GND
P0.2 43 44 DIGITAL GND
P0.1 45 46 DIGITAL GND
P0.0 47 48 DIGITAL GND
+5 VOLTS 49 50 DIGITAL GND
P2
P5.7 1 2 EXT INT 2
P5.6 3 4 DIGITAL GND
P5.5 5 6 DIGITAL GND
P5.4 7 8 DIGITAL GND
P5.3 9 10 DIGITAL GND
P5.2 11 12 DIGITAL GND
P5.1 13 14 DIGITAL GND
P5.0 15 16 DIGITAL GND
P3.7 17 18 DIGITAL GND
P3.6 19 20 DIGITAL GND
P3.5 21 22 DIGITAL GND
P3.4 23 24 DIGITAL GND
P3.3 25 26 DIGITAL GND
P3.2 27 28 DIGITAL GND
P3.1 29 30 DIGITAL GND
P3.0 31 32 DIGITAL GND
P1.7 33 34 DIGITAL GND
P1.6 35 36 DIGITAL GND
P1.5 37 38 DIGITAL GND
P1.4 39 40 DIGITAL GND
P1.3 41 42 DIGITAL GND
P1.2 43 44 DIGITAL GND
P1.1 45 46 DIGITAL GND
P1.0 47 48 DIGITAL GND
+5 VOLTS 49 50 DIGITAL GND
P3

Fig. 2-1 — P2 and P3 50-pin I/O Connector Pin Assignments

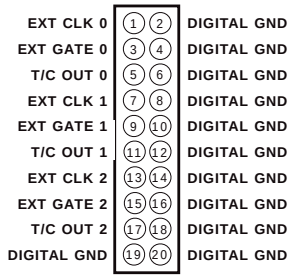



Fig. 2-2 — P6 20-pin I/O Connector Pin Assignments

**Connecting the Digital I/O**

The DIO4806 is designed for direct connection to industry standard opto-22 isolated I/O racks and system modules. Each digital I/O line has a digital ground, as shown in Figure 2-1. For all digital I/O connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to the DIGITAL GND. A cable to provide direct connection to opto-22 systems, the XO50, is available as an accessory from RTD.

**Connecting the Timer/Counter I/O**

External connections to the timer/counters on the DIO4806 can be made by connecting the high side of the external device to the appropriate signal pin on I/O connector P6 and the low side to a P6 DIGITAL GND.

**Connecting the External Interrupt**

The DIO4806 can receive externally generated interrupt signals – EXTINT1, through I/O connector P2, pin 2; and EXTINT2, through I/O connector P3 – and route them to an IRQ channel selected through software or to a port's strobe input pin through header connectors P20 through P22 as described in Chapter 1. Interrupt generation is enabled through software. When interrupts are enabled, a rising or falling edge on the EXTINT line will cause the selected IRQ line to go high, depending on the setting of BA + 16, bit 1, and the IRQ status bit will change from 0 to 1. The pulse applied to the EXTINT pin should have a duration of at least 100 nanoseconds.

## Running the 4806DIAG Diagnostics Program

Now that your board is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 4806DIAG, is included with your example software to help you verify your board's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

# CHAPTER 3

## HARDWARE DESCRIPTION

This chapter describes the features of the DIO4806 hardware.
The major circuits are the digital I/O lines and the timer/counters.

The DIO4806 has two major circuits, the digital I/O lines and the timer/counters. Figure 3-1 shows the block diagram of the board. This chapter describes the hardware which makes up the major circuits.

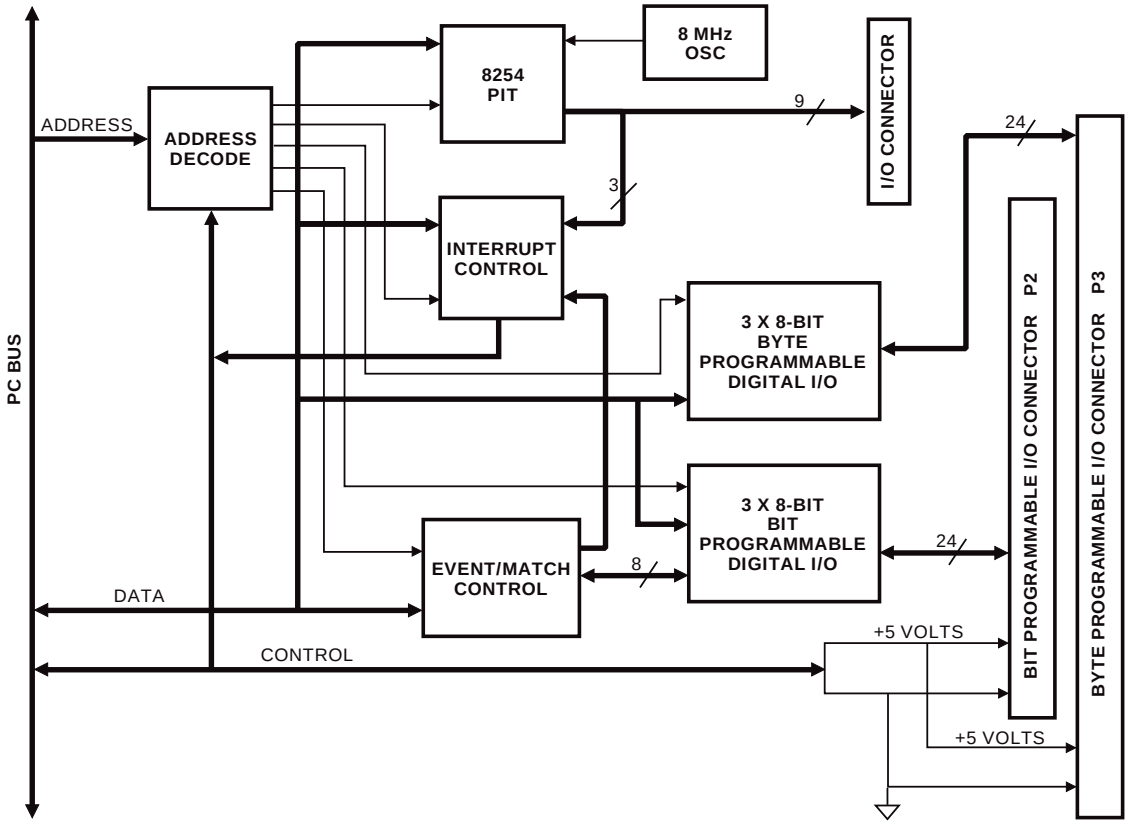



Fig. 3-1 — DIO4806 Block Diagram

## Digital I/O

The 48 digital I/O lines can be used to transfer data between the computer and external devices. Twenty-four lines are bit programmable and 24 lines are byte, or port, programmable.

Ports 0, 2, and 4 each provide eight bit programmable lines which can be independently set for input or output. All three ports support RTD's two Advanced Digital Interrupt modes. An interrupt can be generated when the lines match a programmed value or when any bit changes its current state. A Mask Register lets you monitor selected lines for interrupt generation.

Ports 1, 3, and 5 can be programmed as 8-bit input or output ports.

Chapter 5 details digital I/O operations and Chapter 7 explains digital interrupts.

## Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8-MHz timer/counters to support a wide range of timing and counting functions. Figure 3-2 shows the timer/counter circuitry.

Each 16-bit timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. Each can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in Chapter 4. The command word also lets you set up the mode of operation. The six programmable modes are:

Mode 0 Event Counter (Interrupt on Terminal Count)
Mode 1 Hardware-Retriggerable One-Shot
Mode 2 Rate Generator
Mode 3 Square Wave Mode
Mode 4 Software-Triggered Strobe
Mode 5 Hardware Triggered Strobe (Retriggerable)

These modes are detailed in the 8254 Data Sheet, reprinted from Intel in Appendix C.

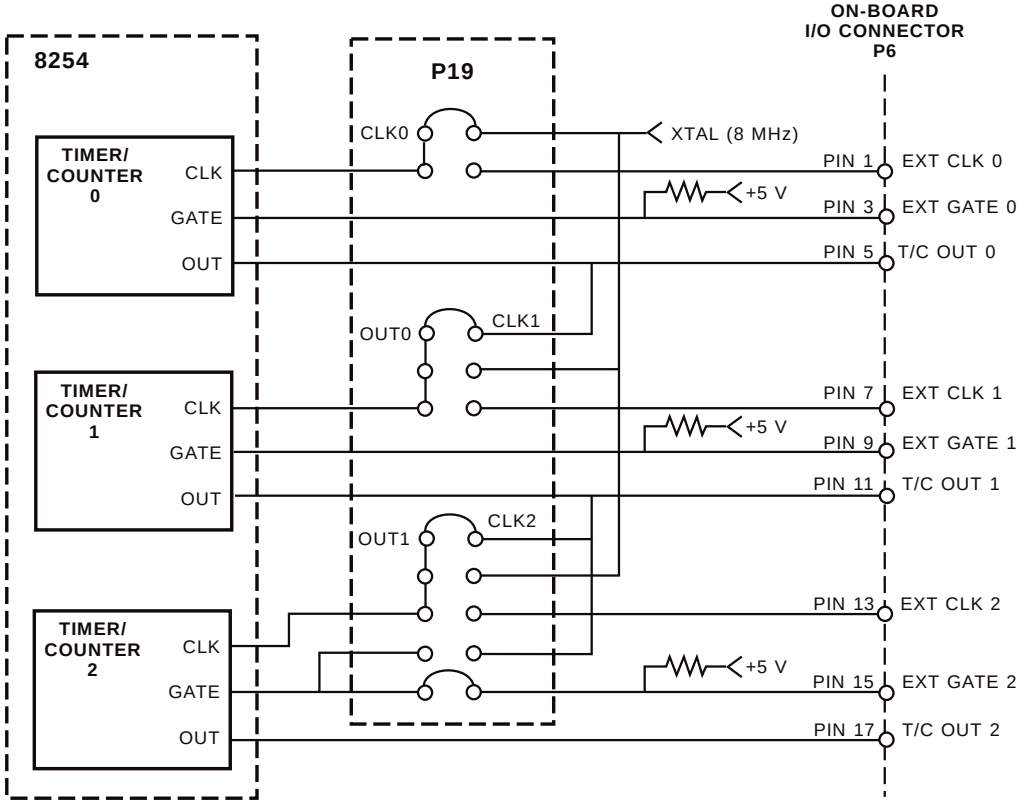



Fig. 3-2 — Timer/Counter Circuit Block Diagram

# CHAPTER 4

## I/O MAPPING

This chapter provides a complete description of the I/O map for the DIO4806, general programming information, and how to set and clear bits in a port.

## Defining the I/O Map

The I/O map for the DIO4806 is shown in Table 4-1 below. As shown, the board occupies 20 consecutive I/O port locations.

To conserve the use of I/O space, the structure of the I/O map is such that some of the registers control what operation you are performing at other addresses. The digital registers you address at BA + 2, 6, and 10 are selected at BA + 3, 7, and 11. This scheme is easily understood once you review the register descriptions on the following pages.

The base address (designated as BA) can be selected using DIP switch S1, located on the edge of the board, as described in Chapter 1, *Board Settings*. This switch can be accessed without removing the board from the computer. The following sections describe the register contents of each address used in the I/O map.

**Table 4-1: DIO4806 I/O Map**

| Register Description | Read Function | Write Function | Address * (Decimal) |
|---|---|---|---|
| Digital I/O Port 0 | Read Port 0 digital input lines | Program Port 0 digital output lines | BA + 0 |
| Digital I/O Port 1 | Read Port 1 digital input lines | Program Port 1 digital output lines | BA + 1 |
| Port 0 Clear/ Direction/Mask/Compare | Clear digital IRQ status flag/read Port 0 control register | Clear digital chip/program Port 0 control register (mode set at BA + 3) | BA + 2 |
| Read Digital IRQ Status/ Set Digital Mode Register | Read digital interrupt status word | Program Port 0/1 digital mode register | BA + 3 |
| Digital I/O Port 2 | Read Port 2 digital input lines | Program Port 2 digital output lines | BA + 4 |
| Digital I/O Port 3 | Read Port 3 digital input lines | Program Port 3 digital output lines | BA + 5 |
| Port 2 Clear/ Direction/Mask/Compare | Clear digital IRQ status flag/read Port 2 control register | Clear digital chip/program Port 2 control register (mode set at BA + 7) | BA + 6 |
| Read Digital IRQ Status/ Set Digital Control Register | Read digital interrupt status word | Program Port 2/3 digital control register | BA + 7 |
| Digital I/O Port 4 | Read Port 4 digital input lines | Program Port 4 digital output lines | BA + 8 |
| Digital I/O Port 5 | Read Port 5 digital input lines | Program Port 5 digital output lines | BA + 9 |
| Port 4 Clear/ Direction/Mask/Compare | Clear digital IRQ status flag/read Port 4 control register | Clear digital chip/program Port 4 control register (mode set at BA + 11) | BA + 10 |
| Read Digital IRQ Status/ Set Digital Control Register | Read digital interrupt status word | Program Port 4/5 digital control register | BA + 11 |
| 8254 TC Counter 0 | Read value in Counter 0 | Load count in Counter 0 | BA + 12 |
| 8254 TC Counter 1 | Read value in Counter 1 | Load count in Counter 1 | BA + 13 |
| 8254 TC Counter 2 | Read value in Counter 2 | Load count in Counter 2 | BA + 14 |
| 8254 Control Word | Reserved | Program counter mode | BA + 15 |
| Clear IRQ/IRQ Enable | Clear interrupt line (P14) | Enable interrupt line (P14) | BA + 16 |
| IRQ Status | Read interrupt status | Reserved | BA + 17 |
| Reserved | Reserved | Reserved | BA + 18 |
| IRQ Channel/Source Select | Reads selected IRQ channel/source | Program IRQ channel & source | BA + 19 |

* BA = Base Address

**BA + 0: Digital I/O Port 0, Bit Programmable Port (Read/Write)**

This port transfers the 8-bit Port 0 bit programmable digital input/output data between the board and external devices. The bits are individually programmed as input or output by writing to the Direction Register at BA + 2. For all bits set as inputs, a read reads the input values and a write is ignored. For all bits set as outputs, a read reads the last value sent out on the line and a write writes the current loaded value out to the line.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |

**BA + 1: Digital I/O Port 1, Byte Programmable Port (Read/Write)**

This port transfers the 8-bit Port 1 digital input or digital output byte between the board and an external device. When Port 1 is set as inputs, a read reads the input values and a write is ignored. When Port 1 is set as outputs, a read reads the last value sent out of the port and a write writes the current loaded value out of the port.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset ), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| I/O7 | I/O6 | I/O5 | I/O4 | I/O3 | I/O2 | I/O1 | I/O0 |

**BA + 2: Read/Program Port 0 Direction/Mask/Compare Registers (Read/Write)**

A read clears the IRQ status flag or provides the contents of one of digital I/O Port 0's three control registers; and a write clears the digital chip or programs one of the three control registers, depending on the setting of bits 0 and 1 at BA + 3. When bits 1 and 0 at BA + 3 are 00, the read/write operations clear the digital IRQ status flag (read) and the digital chip (write). When these bits are set to any other value, one of the three Port 0 registers is addressed.

**Direction Register (BA + 3, bits 1 and 0 = 01):**

**For all bits:**
0 = input
1 = output

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |

This register programs the direction, input or output, of each bit at Port 0.

**Mask Register (BA + 3, bits 1 and 0 = 10):**

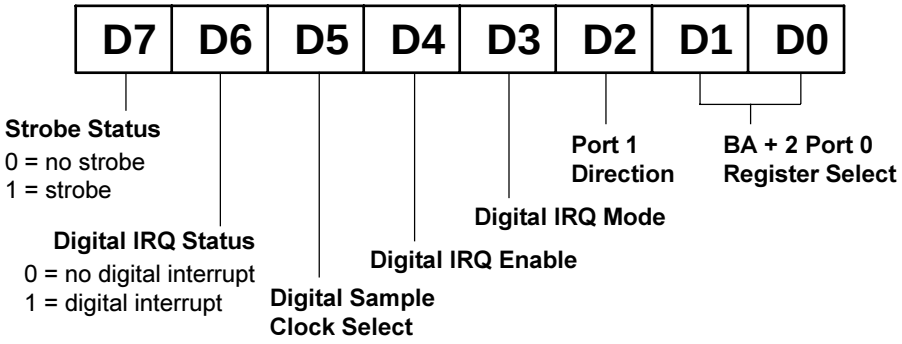
**For all bits:**
0 = bit enabled
1 = bit masked

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |

In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 0 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will not change state, regardless of the digital data written to Port 0. For example, if you set the state of bit 0 low and then mask this bit, the state will remain low, regardless of what you output at Port 0 (an output of 1 will not change the bit's state until the bit is unmasked).

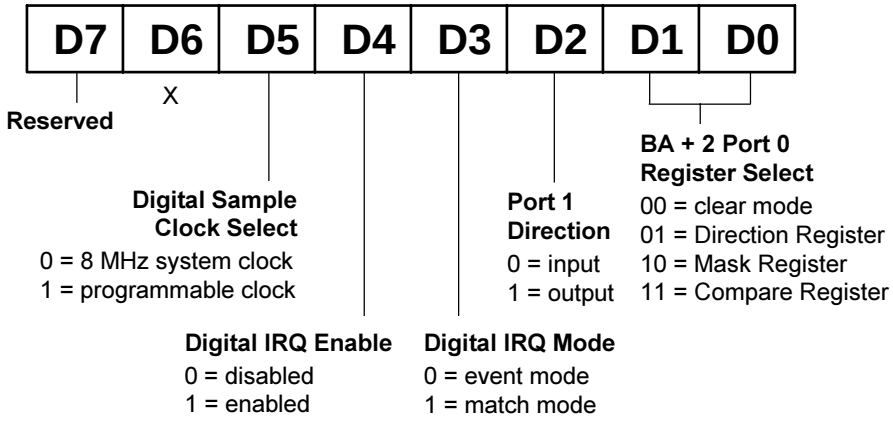
**Compare Register (BA + 3, bits 1 and 0 = 11):**
This register is used for the Advanced Digital Interrupt modes. In the match mode where an interrupt is generated when the Port 0 bits match a loaded value, this register is used to load the bit pattern to be matched at Port 0. Bits can be selectively masked so that they are ignored when making a match. NOTE: Make sure that bit 3 at BA + 3 is set to 1, selecting match mode, BEFORE writing the Compare Register value at this address. In the event mode where an interrupt is generated when any Port 0 bit changes its current state, the value which caused the interrupt is latched at this register and can be read from it. Bits can be selectively masked using the Mask Register so a change of state is ignored on these lines in the event mode.

**BA + 3: Read Digital I/O Status/Program Digital Mode (Read/Write)**




A read shows you whether a digital interrupt has occurred and lets you review the states of the other bits in this register. If bit 6 is high, then a digital interrupt has taken place. This provides the same status information as BA + 17, bit 0.

**Digital Mode Register:**




Bits 0 and 1 – Select the clear mode initiated by a read/write operation at BA + 2 or the Port 0 control register you talk to at BA + 2 (Direction, Mask, or Compare Register).
Bit 2 – Sets the direction of the Port 1 digital lines.
Bit 3 – Selects the digital interrupt mode: event (any Port 0 bit changes state) or match (Port 0 lines match the value programmed into the Compare Register at BA + 2).
Bit 4 – Disables/enables digital interrupts.
Bit 5 – Sets the clock rate at which the digital lines are sampled when in a digital interrupt mode. Available clock sources are the 8 MHz system clock and the output of the 8254 Counter 1 (16-bit programmable clock). When a digital input line changes state, it must stay at the new state for two edges of the clock pulse (62.5 nanoseconds when using the 8 MHz clock) before it is recognized and before an interrupt can be generated. This feature eliminates noise glitches that can cause a false state change on an input line and generate an unwanted interrupt. This feature is detailed in Chapter 5.
Bit 6 – Read only (digital IRQ status).
Bit 7 – Reserved.

**BA + 4: Digital I/O Port 2, Bit Programmable Port (Read/Write)**

This port transfers the 8-bit Port 2 bit programmable digital input/output data between the board and external devices. The bits are individually programmed as input or output by writing to the Direction Register at BA + 6. For all bits set as inputs, a read reads the input values and a write is ignored. For all bits set as outputs, a read reads the last value sent out on the line and a write writes the current loaded value out to the line.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |

**BA + 5: Digital I/O Port 3, Byte Programmable Port (Read/Write)**

This port transfers the 8-bit Port 3 digital input or digital output byte between the board and an external device. When Port 3 is set as inputs, a read reads the input values and a write is ignored. When Port 3 is set as outputs, a read reads the last value sent out of the port and a write writes the current loaded value out of the port.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset ), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| I/O7 | I/O6 | I/O5 | I/O4 | I/O3 | I/O2 | I/O1 | I/O0 |

**BA + 6: Read/Program Port 2 Direction/Mask/Compare Registers (Read/Write)**

A read clears the IRQ status flag or provides the contents of one of digital I/O Port 2's three control registers; and a write clears the digital chip or programs one of the three control registers, depending on the setting of bits 0 and 1 at BA + 7. When bits 1 and 0 at BA + 7 are 00, the read/write operations clear the digital IRQ status flag (read) and the digital chip (write). When these bits are set to any other value, one of the three Port 2 registers is addressed.

**Direction Register (BA + 7, bits 1 and 0 = 01):**

**For all bits:**
0 = input
1 = output

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |

This register programs the direction, input or output, of each bit at Port 2.

**Mask Register (BA + 7, bits 1 and 0 = 10):**

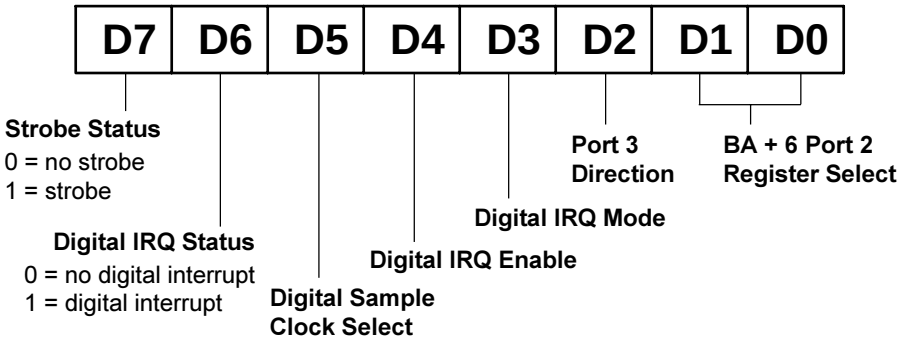
**For all bits:**
0 = bit enabled
1 = bit masked

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |

In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 2 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will not change state, regardless of the digital data written to Port 2. For example, if you set the state of bit 0 low and then mask this bit, the state will remain low, regardless of what you output at Port 2 (an output of 1 will not change the bit's state until the bit is unmasked).

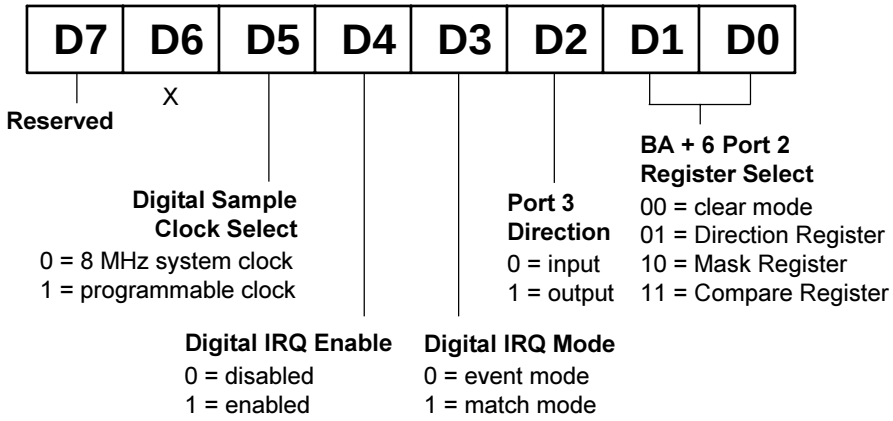
**Compare Register (BA + 7, bits 1 and 0 = 11):**
This register is used for the Advanced Digital Interrupt modes. In the match mode where an interrupt is generated when the Port 2 bits match a loaded value, this register is used to load the bit pattern to be matched at Port 2. Bits can be selectively masked so that they are ignored when making a match. NOTE: Make sure that bit 3 at BA + 7 is set to 1, selecting match mode, BEFORE writing the Compare Register value at this address. In the event mode where an interrupt is generated when any Port 2 bit changes its current state, the value which caused the interrupt is latched at this register and can be read from it. Bits can be selectively masked using the Mask Register so a change of state is ignored on these lines in the event mode.

**BA + 7: Read Digital I/O Status/Program Digital Mode (Read/Write)**




A read shows you whether a digital interrupt has occurred and lets you review the states of the other bits in this register. If bit 6 is high, then a digital interrupt has taken place. This provides the same status information as BA + 17, bit 1.

**Digital Mode Register:**




Bits 0 and 1 – Select the clear mode initiated by a read/write operation at BA + 6 or the Port 2 control register you talk to at BA + 6 (Direction, Mask, or Compare Register).
Bit 2 – Sets the direction of the Port 3 digital lines.
Bit 3 – Selects the digital interrupt mode: event (any Port 2 bit changes state) or match (Port 2 lines match the value programmed into the Compare Register at BA + 6).
Bit 4 – Disables/enables digital interrupts.
Bit 5 – Sets the clock rate at which the digital lines are sampled when in a digital interrupt mode. Available clock sources are the 8 MHz system clock and the output of the 8254 Counter 1 (16-bit programmable clock). When a digital input line changes state, it must stay at the new state for two edges of the clock pulse (62.5 nanoseconds when using the 8 MHz clock) before it is recognized and before an interrupt can be generated. This feature eliminates noise glitches that can cause a false state change on an input line and generate an unwanted interrupt. This feature is detailed in Chapter 5.
Bit 6 – Read only (digital IRQ status).
Bit 7 – Reserved.

**BA + 8: Digital I/O Port 4, Bit Programmable Port (Read/Write)**

This port transfers the 8-bit Port 4 bit programmable digital input/output data between the board and external devices. The bits are individually programmed as input or output by writing to the Direction Register at BA + 10. For all bits set as inputs, a read reads the input values and a write is ignored. For all bits set as outputs, a read reads the last value sent out on the line and a write writes the current loaded value out to the line.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P4.7 | P4.6 | P4.5 | P4.4 | P4.3 | P4.2 | P4.1 | P4.0 |

**BA + 9: Digital I/O Port 5, Byte Programmable Port (Read/Write)**

This port transfers the 8-bit Port 5 digital input or digital output byte between the board and an external device. When Port 5 is set as inputs, a read reads the input values and a write is ignored. When Port 5 is set as outputs, a read reads the last value sent out of the port and a write writes the current loaded value out of the port.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset ), all digital lines are reset to inputs and their corresponding output registers are cleared.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| I/O7 | I/O6 | I/O5 | I/O4 | I/O3 | I/O2 | I/O1 | I/O0 |

**BA + 10: Read/Program Port 4 Direction/Mask/Compare Registers (Read/Write)**

A read clears the IRQ status flag or provides the contents of one of digital I/O Port 4's three control registers; and a write clears the digital chip or programs one of the three control registers, depending on the setting of bits 0 and 1 at BA + 11. When bits 1 and 0 at BA + 11 are 00, the read/write operations clear the digital IRQ status flag (read) and the digital chip (write). When these bits are set to any other value, one of the three Port 4 registers is addressed.

**Direction Register (BA + 11, bits 1 and 0 = 01):**

**For all bits:**
0 = input
1 = output

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P4.7 | P4.6 | P4.5 | P4.4 | P4.3 | P4.2 | P4.1 | P4.0 |

This register programs the direction, input or output, of each bit at Port 4.

**Mask Register (BA + 11, bits 1 and 0 = 10):**

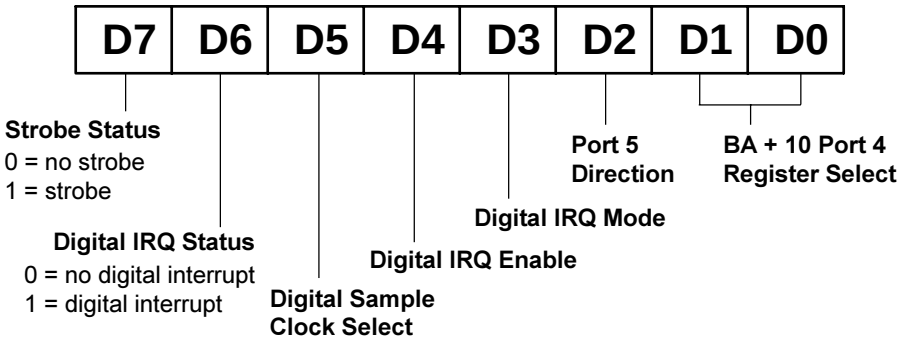
**For all bits:**
0 = bit enabled
1 = bit masked

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P4.7 | P4.6 | P4.5 | P4.4 | P4.3 | P4.2 | P4.1 | P4.0 |

In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 4 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will not change state, regardless of the digital data written to Port 4. For example, if you set the state of bit 0 low and then mask this bit, the state will remain low, regardless of what you output at Port 4 (an output of 1 will not change the bit's state until the bit is unmasked).

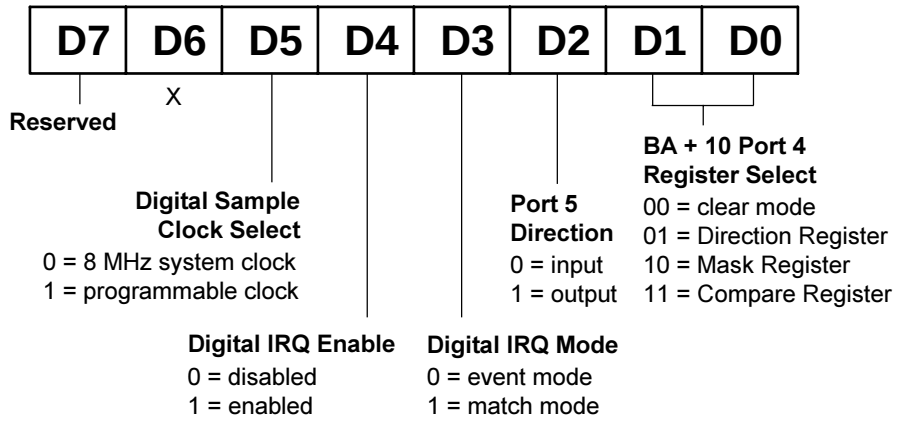
**Compare Register (BA + 11, bits 1 and 0 = 11):**
This register is used for the Advanced Digital Interrupt modes. In the match mode where an interrupt is generated when the Port 4 bits match a loaded value, this register is used to load the bit pattern to be matched at Port 4. Bits can be selectively masked so that they are ignored when making a match. NOTE: Make sure that bit 3 at BA + 11 is set to 1, selecting match mode, BEFORE writing the Compare Register value at this address. In the event mode where an interrupt is generated when any Port 4 bit changes its current state, the value which caused the interrupt is latched at this register and can be read from it. Bits can be selectively masked using the Mask Register so a change of state is ignored on these lines in the event mode.

**BA + 11: Read Digital I/O Status/Program Digital Mode (Read/Write)**




A read shows you whether a digital interrupt has occurred and lets you review the states of the other bits in this register. If bit 6 is high, then a digital interrupt has taken place. This provides the same status information as BA + 27, bit 2.

**Digital Mode Register:**




Bits 0 and 1 – Select the clear mode initiated by a read/write operation at BA + 10 or the Port 4 control register you talk to at BA + 10 (Direction, Mask, or Compare Register).
Bit 2 – Sets the direction of the Port 5 digital lines.
Bit 3 – Selects the digital interrupt mode: event (any Port 4 bit changes state) or match (Port 4 lines match the value programmed into the Compare Register at BA + 10).
Bit 4 – Disables/enables digital interrupts.
Bit 5 – Sets the clock rate at which the digital lines are sampled when in a digital interrupt mode. Available clock sources are the 8 MHz system clock and the output of the 8254 Counter 1 (16-bit programmable clock). When a digital input line changes state, it must stay at the new state for two edges of the clock pulse (62.5 nanoseconds when using the 8 MHz clock) before it is recognized and before an interrupt can be generated. This feature eliminates noise glitches that can cause a false state change on an input line and generate an unwanted interrupt. This feature is detailed in Chapter 5.
Bit 6 – Read only (digital IRQ status).
Bit 7 – Reserved.

**BA + 12: 8254 Timer/Counter 0 (Read/Write)**

This address is used to read/write timer/counter 0. A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

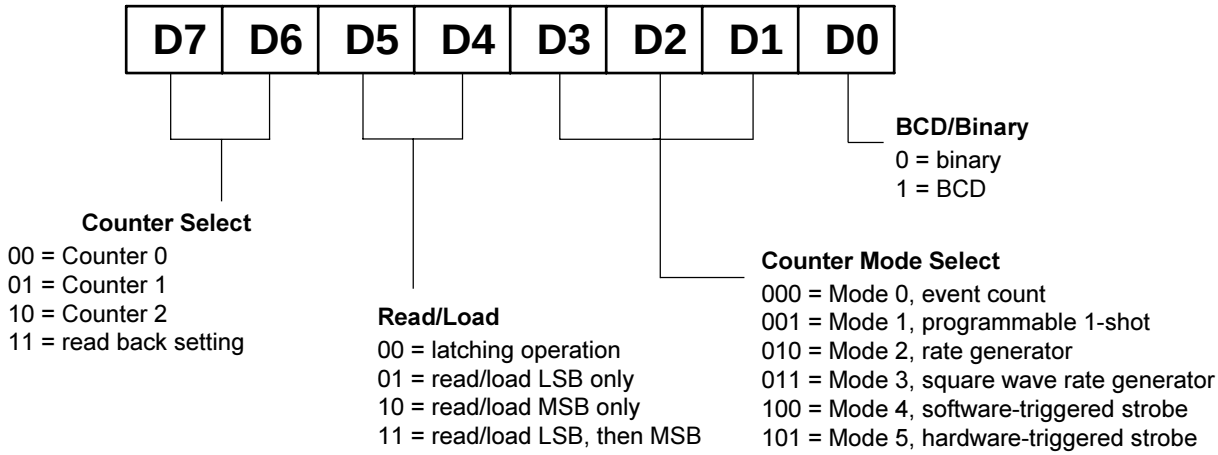
**BA + 13: 8254 Timer/Counter 1 (Read/Write)**

This address is used to read/write timer/counter 1. A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

**BA + 14: 8254 Timer/Counter 2 (Read/Write)**

This address is used to read/write timer/counter 2. A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

**BA + 15: 8254 Timer/Counter Control Word (Write Only)**

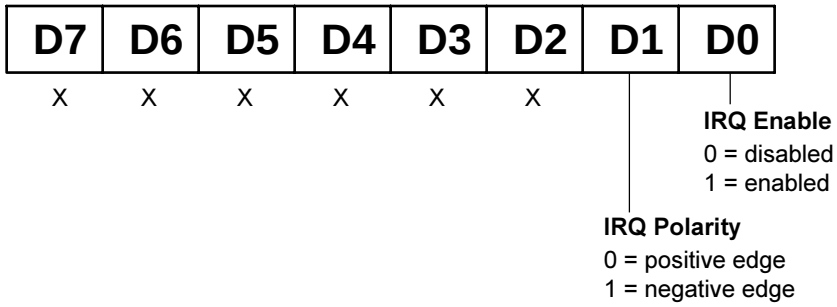

D7 D6 D5 D4 D3 D2 D1 D0
BCD/Binary
0 = binary
1 = BCD
Counter Select
00 = Counter 0
01 = Counter 1
10 = Counter 2
11 = read back setting
Read/Load
00 = latching operation
01 = read/load LSB only
10 = read/load MSB only
11 = read/load LSB, then MSB
Counter Mode Select
000 = Mode 0, event count
001 = Mode 1, programmable 1-shot
010 = Mode 2, rate generator
011 = Mode 3, square wave rate generator
100 = Mode 4, software-triggered strobe
101 = Mode 5, hardware-triggered strobe

This address is used to write to the control register for the 8254. The control word is defined above.

**BA + 16: Clear IRQ/IRQ Enable (Read/Write)**

A read clears the board's IRQ status flag at BA + 17, bit 6.

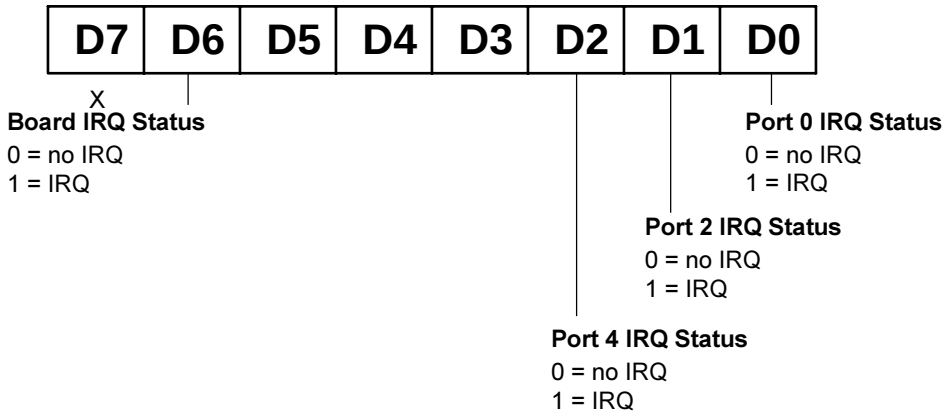
**IRQ Enable Register:**




A write enables software programmable interrupts at BA + 19 and selects whether the interrupt will occur on the positive (rising) edge or negative (falling) edge of the pulse.

**BA + 17: IRQ Status (Read Only)**

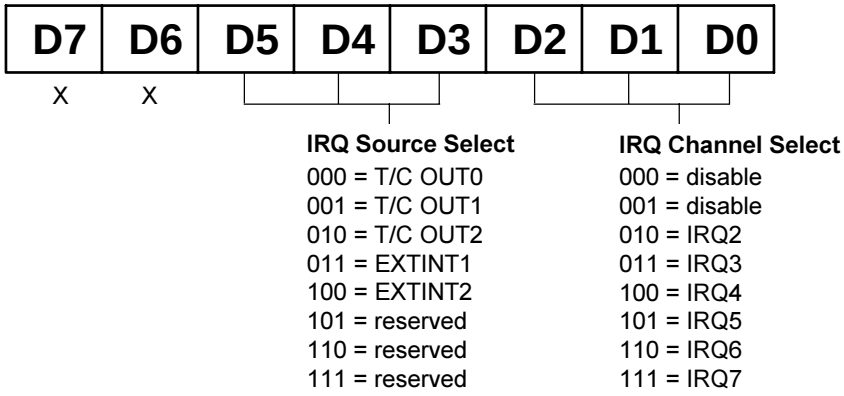
A read shows the status of each of the six Advanced Digital Interrupt circuits (bits 0 through 5) and the board's interrupt circuit (bit 6) so that you can determine which circuit generated an interrupt.




**BA + 18: Reserved**

**BA + 19: IRQ Channel/Source Select (Read/Write)**

A read lets you review the IRQ source and channel.




A write programs the IRQ source and channel as shown above.

## Programming the DIO4806

This section gives you some general information about programming and the DIO4806.

The board is programmed by reading from and writing to the correct I/O port locations. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

| Language | Read | Write |
|---|---|---|
| BASIC | Data=INP(Address) | OUT Address,Data |
| Turbo C | Data=inportb(Address) | outportb(Address,Data) |
| Turbo Pascal | Data:=Port[Address] | Port[Address]:=Data |
| Assembly | mov dx,Address<br>in al,dx | mov dx,Address<br>mov al,Data<br>out dx,al |

In addition to being able to read/write the I/O ports on the DIO4806, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with C, Pascal, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

| Language | Modulus | Integer Division | AND | OR |
|---|---|---|---|---|
| C | %<br>a = b % c | /<br>a = b / c | &<br>a = b & c | \|<br>a = b \| c |
| Pascal | MOD<br>a := b MOD c | DIV<br>a := b DIV c | AND<br>a := b AND c | OR<br>a := b OR c |
| BASIC | MOD<br>a = b MOD c | \<br>a = b \ c | AND<br>a = b AND c | OR<br>a = b OR c |

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with the DIO4806!**

## Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation. Note that most registers in the DIO4806 cannot be read back; therefore, you must save the value in your program.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

**Example:** Clear bit 5 in a port. Read in the current value of the port, AND it with 223 ($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V_SAVE = V_SAVE AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

**Example:** Set bit 3 in a port. Read in the current value of the port, OR it with 8 ($8 = 2^3$), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V_Save = V_Save OR 8;
Port[PortAddress] := V_Save;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 - (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

**Example:** Clear bits 2, 4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v_save = v_save & 171;
outportb(port_address, v_save);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

**Example:** Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov al, v_save
or al, 168
mov dx, PortAddress
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

**Example:** Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v_save = v_save & 199;
v_save = v_save | 40;
outportb(port_address, v_save);
```

**A final note:** Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 ($2^5$) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 *is* already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

# CHAPTER 5

## DIGITAL I/O

This chapter explains the bit programmable and port programmable digital I/O circuitry on the DIO4806.

The DIO4806 has 48 buffered TTL/CMOS digital I/O lines available for digital control applications. These lines are grouped in six 8-bit ports. Each of the eight bits in Ports 0, 2, and 4 can be independently programmed as input or output. Ports 1, 3, and 5 can be programmed as an 8-bit input or output port.

## Ports 0, 2 and 4: Bit Programmable Digital I/O

The eight Port 0, Port 2, and Port 4 digital lines are individually set for input or output by writing to the respective Direction Registers at BA + 2, BA + 6, and BA + 10. The input lines are read and the output lines are written at BA + 0, BA + 4, and BA + 8.

**Direction Register:**

**For all bits:**
0 = input
1 = output

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| PX.7 | PX.6 | PX.5 | PX.4 | PX.3 | PX.2 | PX.1 | PX.0 |

**Advanced Digital Interrupts: Mask and Compare Registers**

The Ports 0, 2, and 4 bits support two Advanced Digital Interrupt modes. An interrupt can be generated when the data read at the port matches the value loaded into its Compare Register. This is called a match interrupt. NOTE: Make sure that the port's Digital IRQ Mode bit is set to 1, selecting match mode, BEFORE writing the Compare Register value for the port. An interrupt also can be generated whenever any bit changes state. This is an event interrupt. For either interrupt, bits can be masked by setting the corresponding bit in the port's Mask Register high. In a digital interrupt mode, this masks out selected bits when monitoring the bit pattern for a match or event. In normal operation where the Advanced Digital Interrupt mode is not activated, the Mask Register can be used to preserve a bit's state, regardless of the digital data written to the port.

When using event interrupts, you can determine which bit caused an event interrupt to occur by reading the contents latched into the Compare Register.

## Ports 1, 3, and 5: Port Programmable Digital I/O

The directions of the eight Port 1, Port 3, and Port 5 digital lines are programmed at bit 2 at address locations BA + 3, BA + 7, and BA + 11, respectively. These lines are configured as all inputs or all outputs, with their states read and written at BA + 1 (Port 1), BA + 5 (Port 3), and BA + 9 (Port 5).

## Resetting the Digital Circuitry

When a digital chip clear is issued, all of the digital I/O lines are set up as inputs and their corresponding output registers are cleared.

## Strobing Data into Ports 0, 2, and 4

When not in an Advanced Digital Interrupt mode, external data can be strobed into Ports 0, 2, and/or 4 by using the EXTINT1 signal at P2-2, or EXTINT2 signal at P3-2. The EXTINT signal selected is jumpered for a port at header connectors P20 through P22, as described in Chapter 1. The data strobed in can be read from the port's Compare Register.

# CHAPTER 6

## TIMER/COUNTERS

This chapter explains the 8254 timer/counter circuit on the DIO4806.

An 8254 programmable interval timer provides three 16-bit, 8-MHz timers for timing and counting functions such as frequency measurement, event counting, and interrupts. These timer/counters can be configured in a number of ways to support your application. Figure 6-1 shows a block diagram of the timer/counter circuitry.

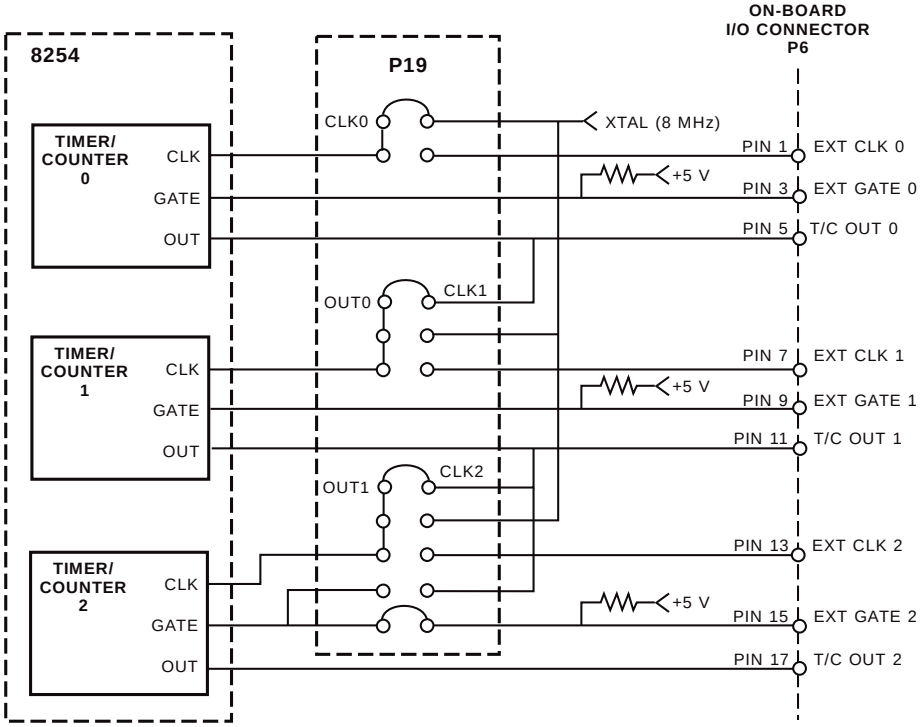



Fig. 6-1 — 8254 Timer/Counter Circuit Block Diagram

Each timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. They can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in the I/O map discussion in Chapter 4.

The timer/counter outputs are available at P6 where they can be used for interrupt generation, as an A/D trigger, or for timing and counting functions.

The timers can be programmed to operate in one of six modes, depending on your application. The following paragraphs briefly describe each mode.

**Mode 0, Event Counter (Interrupt on Terminal Count).** This mode is typically used for event counting. While the timer/counter counts down, the output is low, and when the count is complete, it goes high. The output stays high until a new Mode 0 control word is written to the timer/counter.

**Mode 1, Hardware-Retriggerable One-Shot.** The output is initially high and goes low on the clock pulse following a trigger to begin the one-shot pulse. The output remains low until the count reaches 0, and then goes high and remains high until the clock pulse after the next trigger.

**Mode 2, Rate Generator.** This mode functions like a divide-by-N counter and is typically used to generate a real-time clock interrupt. The output is initially high, and when the count decrements to 1, the output goes low for one clock pulse. The output then goes high again, the timer/counter reloads the initial count, and the process is repeated. This sequence continues indefinitely.

**Mode 3, Square Wave Mode.** Similar to Mode 2 except for the duty cycle output, this mode is typically used for baud rate generation. The output is initially high, and when the count decrements to one-half its initial count, the output goes low for the remainder of the count. The timer/counter reloads and the output goes high again. This process repeats indefinitely.

**Mode 4, Software-Triggered Strobe.** The output is initially high. When the initial count expires, the output goes low for one clock pulse and then goes high again. Counting is "triggered" by writing the initial count.

**Mode 5, Hardware Triggered Strobe (Retriggerable).** The output is initially high. Counting is triggered by the rising edge of the gate input. When the initial count has expired, the output goes low for one clock pulse and then goes high again.

Appendix C provides the 8254 data sheet.

# CHAPTER 7

## INTERRUPTS

This chapter explains programmable interrupts, digital interrupts, and basic interrupt programming techniques.

The DIO4806 has five interrupt circuits which can generate interrupts on any IRQ channel 2 through 7.

## Software Programmable Interrupts

The DIO4806 circuitry has five software selectable interrupt sources which can be selected at BA + 19, bits 3 through 5, as described in Chapter 4.

To use these interrupts, an interrupt source must be selected at BA + 19, an interrupt channel must be enabled at BA + 19, bits 0 through 2, and the IRQ enable must be set high (BA + 16, bit 0). BA + 16, bit 1 sets the polarity of the interrupt.

## Advanced Digital Interrupts

Each bit programmable digital I/O port supports two Advanced Digital Interrupt modes, event mode or match mode. These modes are used to monitor input lines for state changes. The mode is selected in the port's Control Register, bit 3 and enabled in the Control Register, bit 4.

### Event Mode

When enabled, this mode samples the port's input lines at a specified clock rate (using the 8 MHz system clock or a programmable clock in the timer/counter programmed at bit 5 of the port's Digital Mode Register), looking for a change in state in any one of the eight bits. When a change of state occurs, an interrupt is generated and the input pattern is latched into the Compare Register. You can read the contents of this register to see which bit caused the interrupt to occur. Bits can be masked and their state changes ignored by programming the Mask Register.

### Match Mode

When enabled, this mode samples the port's input lines at a specified clock rate (using the 8 MHz system clock or a programmable clock in the timer/counter programmed at bit 5 of the port's Digital Mode Register) and compares all input states to the value programmed in the Compare Register. When the states of all of the lines match the value in the Compare Register, an interrupt is generated. Bits can be masked and their states ignored by programming the Mask Register. NOTE: Make sure that the port's Digital IRQ Mode bit is set to 1, selecting match mode, BEFORE writing the Compare Register value for the port.

### Sampling Digital Lines for Change of State

In the Advanced Digital Interrupt modes, the digital lines are sampled at a rate set by the 8 MHz system clock or the clock programmed in the timer/counter programmed at bit 5 of the port's Digital Mode Register. With each clock pulse, the digital circuitry looks at the state of the next bit. To provide noise rejection and prevent erroneous interrupt generation because of noise spikes on the digital lines, a change in the state of any bit must be seen for two edges of a clock pulse to be recognized by the circuit. Figure 7-1 shows a diagram of this circuit.

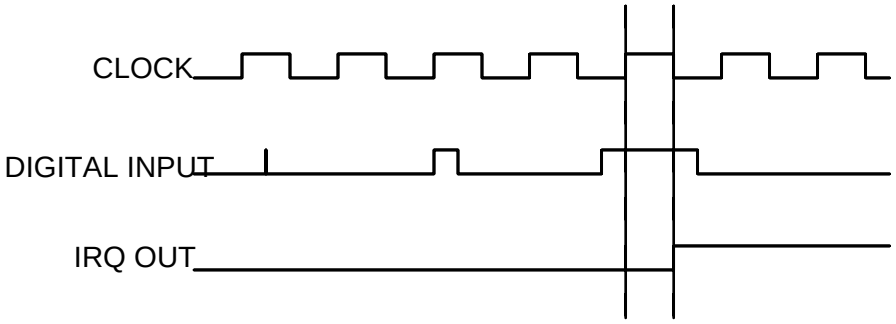



Fig. 7-1 — Digital Interrupt Timing Diagram

## Selecting the Interrupt Channel

The IRQ channel is selected by programming the IRQ channel at BA + 19, bits 0 through 2.

To determine which interrupt source has generated an interrupt, you must check bits 0 through 2 of the status word read at BA + 17. Then service the interrupt that has occurred and clear the interrupt (the software programmable interrupt is cleared by reading BA + 16, and the digital interrupts are cleared by setting bits 1 and 0 in the corresponding port's Control Register and performing a read at the Direction/Mask/Compare Register address).

## Basic Programming For Interrupt Handling

### What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard 'interrupts' the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your DIO4806 can interrupt the processor when a variety of conditions are met. By using these interrupts, you can write software that effectively deals with real world events.

### Interrupt Request Lines

To allow different peripheral devices to generate interrupts on the same computer, the PC bus has eight different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by the PC's interrupt controller. The interrupt controller checks to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is based on the number of the IRQ; IRQ0 has the highest priority, IRQ1 is second-highest, and so on through IRQ7, which has the lowest. Many of the IRQs are used by the standard system resources. IRQ0 is used by the system timer, IRQ1 is used by the keyboard, IRQ3 by COM2, IRQ4 by COM1, and IRQ6 by the disk drives. Therefore, it is important for you to know which IRQ lines are available in your system for use by the module.

### 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in the PC is the 8259 Programmable Interrupt Controller. To use interrupts, you need to know how to read and set the 8259's interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to the 8259.

#### - Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. If a bit is **set** (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is **clear** (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR is programmed through port 21H.

| IRQ7 | IRQ6 | IRQ5 | IRQ4 | IRQ3 | IRQ2 | IRQ1 | IRQ0 |
|---|---|---|---|---|---|---|---|

I/O Port 21H

**For all bits:**
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

#### - End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 interrupt controller must be notified. This is done by writing the value 20H to I/O port 20H.

### What Exactly Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DIO9606), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

### Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the INTRPTS source code included on your DIO4806 program disk for a better understanding of interrupt program development.

### Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the stack **BEFORE** you do anything else. Second, just before exiting your ISR, you must clear the interrupt status flag of the DIO4806 and write an end-of-interrupt command to the 8259 controller. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by interrupt programming, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers **do not** automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

**NOTE:** If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.

There are a few cautions you must consider when writing your ISR. The most important is, **do not use any DOS functions or routines that call DOS functions from within an ISR**. DOS is **not** reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Clear the interrupt status flag for the source which caused the interrupt:
  - Clear software programmable interrupt status flag by reading BA + 16.
  - Clear the Port 0 digital interrupt flag by setting bits 1 and 0 at BA + 3 to 00 and reading BA + 2.
  - Clear the Port 2 digital interrupt flag by setting bits 1 and 0 at BA + 7 to 00 and reading BA + 6.
  - Clear the Port 4 digital interrupt flag by setting bits 1 and 0 at BA + 11 to 00 and reading BA + 10.
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H.
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like. Only the clear interrupt command sequence for the source which caused the interrupt needs to be included:

**In C:**

```
void interrupt ISR(void)
{
    /* Your code goes here. Do not use any DOS functions! */
    inportb(BaseAddress + 16);              /* Clear software programmable interrupt */
    outportb(BaseAddress + 3, 0);           /* Set Port 0 digital I/O clear mode */
    inportb(BaseAddress + 2);               /* Clear Port 0 digital interrupt */
        .
        .
    outportb(BaseAddress + 11, 0);          /* Set Port 4 digital I/O clear mode */
    inportb(BaseAddress + 10);              /* Clear Port 4 digital interrupt */
    outportb(0x20, 0x20);                   /* Send EOI command to 8259 */
}
```

**In Pascal:**

```
Procedure ISR; Interrupt;
begin
    { Your code goes here. Do not use any DOS functions! }
    c := Port[BaseAddress + 16];            { Clear jumper selectable interrupt }
    Port[BaseAddress + 3] := 0;             { Set Port 0 digital I/O clear mode }
    c := Port[BaseAddress + 2];             { Clear Port 0 digital interrupt }
        .
        .
    Port[BaseAddress + 11] := 0;            { Set Port 4 digital I/O clear mode }
    c := Port[BaseAddress + 10];            { Clear Port 4 digital interrupt }
    Port[$20] := $20;                       { Send EOI command to 8259 }
end;
```

### Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the DIO4806 will be using IRQ3, you should save the value of interrupt vector 11.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21H and **set** the bit that corresponds to your IRQ (remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this chapter for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21H.

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vector 8 is for IRQ0, vector 9 is for IRQ1, and so on.

If you need to program the source of your interrupts, do that next. For example, if you are using the programmable interval timer to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

### Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in when your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H. Restore the interrupt vector that was saved at startup with either DOS function 35H (get interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

### Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15, even though the corresponding IRQs are numbered 0 through 7.
- Two of the most common mistakes when writing an ISR are forgetting to clear the interrupt status of the DIO4806 and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

# APPENDIX A

## DIO4806 SPECIFICATIONS

## DIO4806 Characteristics Typical @ 25° C

### Interface

Switch-selectable base address, I/O mapped
Software programmable interrupts

### Digital I/O

Number of lines ..... 24 bit programmable & 24 port programmable
Isource ..... -12 mA
Isink ..... 24 mA

### Timer/Counters ..... CMOS 82C54

Three 16-bit down counters
6 programmable operating modes
Counter input source ..... External clock (8 MHz, max) or on-board 8-MHz clock
Counter outputs ..... Available externally; used as PC interrupts
Counter gate source ..... External gate or always enabled

### Miscellaneous Inputs/Outputs (PC bus-sourced)

+5 volts, ground

### Power Requirements

+5V @ 250 mA = 1.25W typical

### Connectors

P2 and P3: 50-pin right angle header
P6: 12-pin box header

### Environmental

Operating temperature ..... 0 to +70°C
Storage temperature ..... -40 to +85°C
Humidity ..... 0 to 90% non-condensing

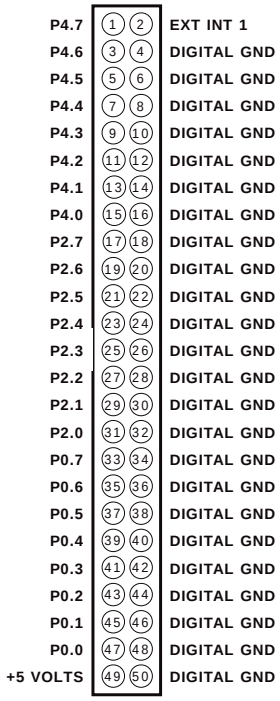
### Size

3.875"H x 13.20"W (99mm x 335mm)

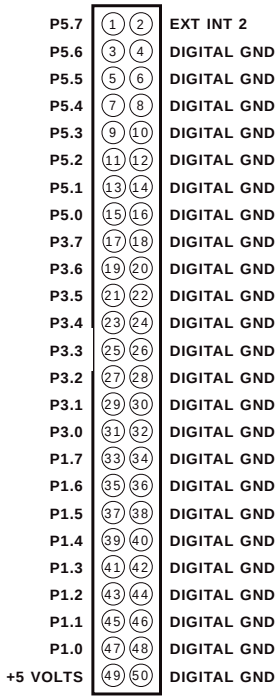
# APPENDIX B

## CONNECTOR PIN ASSIGNMENTS

**P2 Connector:**

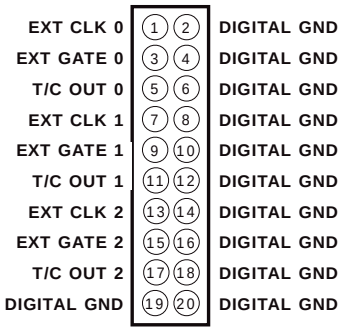

P4.7 1 2 EXT INT 1
P4.6 3 4 DIGITAL GND
P4.5 5 6 DIGITAL GND
P4.4 7 8 DIGITAL GND
P4.3 9 10 DIGITAL GND
P4.2 11 12 DIGITAL GND
P4.1 13 14 DIGITAL GND
P4.0 15 16 DIGITAL GND
P2.7 17 18 DIGITAL GND
P2.6 19 20 DIGITAL GND
P2.5 21 22 DIGITAL GND
P2.4 23 24 DIGITAL GND
P2.3 25 26 DIGITAL GND
P2.2 27 28 DIGITAL GND
P2.1 29 30 DIGITAL GND
P2.0 31 32 DIGITAL GND
P0.7 33 34 DIGITAL GND
P0.6 35 36 DIGITAL GND
P0.5 37 38 DIGITAL GND
P0.4 39 40 DIGITAL GND
P0.3 41 42 DIGITAL GND
P0.2 43 44 DIGITAL GND
P0.1 45 46 DIGITAL GND
P0.0 47 48 DIGITAL GND
+5 VOLTS 49 50 DIGITAL GND

**P3 Connector:**


P5.7 1 2 EXT INT 2
P5.6 3 4 DIGITAL GND
P5.5 5 6 DIGITAL GND
P5.4 7 8 DIGITAL GND
P5.3 9 10 DIGITAL GND
P5.2 11 12 DIGITAL GND
P5.1 13 14 DIGITAL GND
P5.0 15 16 DIGITAL GND
P3.7 17 18 DIGITAL GND
P3.6 19 20 DIGITAL GND
P3.5 21 22 DIGITAL GND
P3.4 23 24 DIGITAL GND
P3.3 25 26 DIGITAL GND
P3.2 27 28 DIGITAL GND
P3.1 29 30 DIGITAL GND
P3.0 31 32 DIGITAL GND
P1.7 33 34 DIGITAL GND
P1.6 35 36 DIGITAL GND
P1.5 37 38 DIGITAL GND
P1.4 39 40 DIGITAL GND
P1.3 41 42 DIGITAL GND
P1.2 43 44 DIGITAL GND
P1.1 45 46 DIGITAL GND
P1.0 47 48 DIGITAL GND
+5 VOLTS 49 50 DIGITAL GND

| P2 & P3 Mating Connector Part Numbers | |
|---|---|
| **Manufacturer** | **Part Number** |
| AMP | 1-746094-0 |
| 3M | 3425-7650 |

**P6 Connector:**




| P6 Mating Connector Part Numbers | |
|---|---|
| **Manufacturer** | **Part Number** |
| AMP | 1-746094-4 |

# APPENDIX C

## COMPONENT DATA SHEETS

# Intel 82C54 Programmable Interval Timer
# Data Sheet Reprint

# APPENDIX D

## WARRANTY

# LIMITED WARRANTY

Real Time Devices, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from REAL TIME DEVICES. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, REAL TIME DEVICES will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to REAL TIME DEVICES. All replaced parts and products become the property of REAL TIME DEVICES. **Before returning any product for repair, customers are required to contact the factory for an RMA number.**

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by REAL TIME DEVICES, “acts of God” or other contingencies beyond the control of REAL TIME DEVICES), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN REAL TIME DEVICES. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND REAL TIME DEVICES EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER’S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL REAL TIME DEVICES BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

| DIO4806 User Settings | |
|---|---|
| **Base I/O Address:** | |
| (hex) | (decimal) |